# Fast Predictive Control of Micro Controller's Energy-Performance Tradeoff

Sylvain Durand, Nicolas Marchand

# Fast Predictive Control of Micro Controller's Energy-Performance Tradeoff

Sylvain Durand and Nicolas Marchand

***Abstract*— A two voltage level electronic device is interesting because the clock frequency and the supply voltage level could be reduced in order to decrease the energy consumption. However, these two quantities have to be controlled respecting certain rules, and decreasing them leads to a reduced computational speed. In this paper a control architecture is proposed to deal with this power-performance tradeoff. First, a fast predictive control technic gives the best computational speed set point to minimize the penalizing high voltage running time. Then, the frequency and the supply voltage are controlled together in order that the measured speed tracks this set point. Finally, the proposal clearly gives an important reduction of the energy consumption. Moreover, the control strategy is robust to process variability and therefore suitable for 45nm, 32nm or smaller implementations.**

## I. INTRODUCTION

An energy-performance tradeoff is required in many embedded electronic systems. Actually, three power consumption sources exist in CMOS circuits [4], which could be sorted into a dynamic consumption from switching of electrical gates and a static consumption from short circuit and leakage currents:

$$\begin{aligned} P &= P_{switching} + P_{short\ circuit} + P_{leakage} \\ P &= K_{dyn} f_{clk} V_{dd}^2 + K_{sc} f_{clk} V_{dd} + K_{leak} V_{dd} \end{aligned} \quad (1)$$

From this relation, it seems that the consumption could be reduced by decreasing $V_{dd}$, i.e. the supply voltage, or $f_{clk}$, i.e. the clock frequency. However, decreasing only the frequency will decrease the power consumption and results in a slower running task but the total energy consumption will remain unchanged [14]. The voltage has hence to be reduce in order to decrease the energy consumption. Furthermore, the supply voltage is the dominant term especially because the dynamic power is the most important part in (1). In other words, decreasing the voltage will almost quadratically decrease the energy consumption. Unfortunately, this drop will decrease the computational speed because of the propagation delay of transistors, i.e. $T_d$, which is seriously increasing as $V_{dd}$ approaches the threshold voltage of the device $V_t$:

$$T_d \propto \frac{V_{dd}}{(V_{dd} - V_t)^2}$$

Controlling the supply voltage is hence a power-delay tradeoff: the power consumption decreases while the delay increases. That is why the supply voltage and the clock frequency have to be controlled together to guarantee the critical path (the longest electrical path a signal can travel to go from a point to another of the circuit): if the frequency is not sufficient the signal could not travel between two clock signals and the result of the calculation made by the system may be wrong. Therefore the critical path delay has to be lower than the inverse of the clock frequency else the system will not correctly work [13]:

$$T_{d\ critical\ path} < \frac{1}{f_{clk}}$$

Clearly, it is required to decrease the clock frequency before decreasing the supply voltage and, respectively, increase the supply voltage before increasing the clock frequency. This principle is needed in all systems to guarantee the critical path: either with an hardware solution like some delay lines [5], [6], or with a software technic at least.

A good consumption-performance tradeoff could be achieved using a commonly used approach in embedded systems: the Dynamic Voltage and Frequency Scaling (*DVFS* or *DVS*). This method consists in adapting the voltage and the frequency to the computational load and leads up to an important energy consumption reduction (regarding the application) [12]. Furthermore, it turns out that different kinds of tasks exist, such as intensive computational tasks, background tasks or processor idle [2] and it seems that most of the applications could run with a reduced voltage [3].

Several behaviors are accepted to minimize the energy consumption. Firstly, each task has to be considered independently and its execution time has to fit with the deadline. Moreover, selecting some suitable voltage levels leads to a drastic energy reduction even if the number of levels is very small [9]. The supply voltage has to be reduced as much as possible and the frequency clock adapted to the computational load to minimize the energy consumption [13].

In this paper, a control strategy that minimizes the energy consumption as much as possible while guaranteeing a computational speed performance is proposed. In the following section, the system architecture of a micro-controller is given. Its model is the derived recalling elementary relationships in electronic devices. In section III, a predictive control technic is detailed to build a computational speed set point in order to minimize the energy consumption of each task. Then, in section IV a simple frequency and voltage level control is presented to track this reference. Finally the controller is simulated in section V and the robustness is tested in particular in the case of high dispersion phenomena like the one arising in 45 nm and smaller technologies.

S. Durand is with NeCS Project-Team, INRIA - GIPSA-lab - CNRS, Grenoble, France, sylvain.durand@inrialpes.fr

N. Marchand is with NeCS Project-Team, INRIA - GIPSA-lab - CNRS, Grenoble, France, nicolas.marchand@gipsa-lab.inpg.fr

## II. SYSTEM ARCHITECTURE

The system architecture is shown on Figure 1.

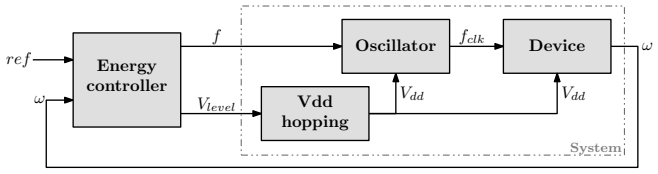


Fig. 1. System architecture

***Device*** is the system to control (a processor or a system on chip for example). It usually runs at nominal supply voltage and constant clock frequency but these quantities will now dynamically vary in order to reduce the energy consumption. That is possible introducing a closed-loop with a controller to monitor the activity of the device (its computational speed $\omega$) and to adapt the supply voltage and the clock frequency regarding the computational load $ref$ provided by the operating system for each task. The equation used for this block is $\omega = \alpha(V_{dd}) \cdot f_{clk} + \beta(V_{dd})$, which can be simplified as $\omega = \alpha \cdot f_{clk} + \beta$ due to the small impact of the voltage on the variables [15].

***Oscillator & Vdd-hopping*** are the two actuators used in some DVFS systems. They respectively provide the clock frequency and the supply voltage to the device.

- The oscillator could be a ring oscillator with the relation $f_{clk} = \gamma \cdot f \cdot V_{dd}$ [8].
- The Vdd-hopping principle is described in [1]. Two voltage levels are available ($V_{low}$ and $V_{high}$) and the one or the other could be achieved (with a certain transition time and dynamics that depends upon the internal controller of the Vdd-hopping, see [1] for further details) regarding the $V_{level}$ input signal: $V_{level} = level_{low}$ to require the low voltage and respectively $level_{high}$ for the high voltage. To sum up, $V_{dd} = f(V_{level})$.

Afterwards, the group "oscillator + Vdd-hopping + device" will be called the system. The system model is given by $\omega = \alpha \cdot \gamma \cdot f \cdot V_{dd} + \beta$, which can finally be approximated by an affine function, as depicted with the following equation:

$$\omega = \sigma \cdot f \cdot V_{dd} \qquad \text{with } \sigma \simeq \alpha \cdot \gamma \tag{2}$$

***Energy controller*** has to provide the control signals to the actuators. Actually, the energy controller can be divided into two parts, as depicted on Figure 2:

- A ***computational speed controller*** which provides the computational speed set point $\omega_{sp}$. Thus, from some task information given by the operating system, this controller chooses the best speed reference in order to minimize the energy consumption while guaranteeing the computational performance.
- A ***frequency and voltage level controller*** which fits the measured speed $\omega$ with the desired one $\omega_{sp}$, by adapting the frequency $f$ and the voltage level $V_{level}$.

The control strategy consists in a feedback loop with the measured computational speed $\omega$ for both parts of the controller (see [7] for different architectures). In the next two sections, we will first deal with the computational speed controller and describe the predictive control law used to build an energy efficient speed set point and then we will detail the frequency and voltage level controller.

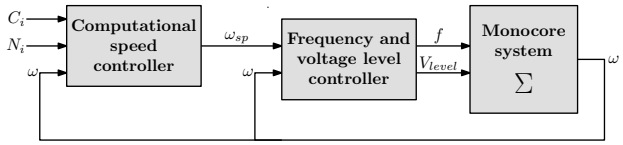


Fig. 2. Energy controller architecture: a computational speed controller plus a frequency and voltage level controller

## III. COMPUTATIONAL SPEED CONTROL

To define the speed set point $\omega_{sp}$ (in number of instructions by second) some task information are required. Indeed, for each task $T_i$ the operating system provides data to the controller: the computational load and the time before the task has to be executed, which are respectively the number of instructions $C_i$ and the deadline $N_i$ (i.e. the inputs on Figure 2). In fact, the remaining available time to execute the task, i.e. the laxity $L_i$, would be used instead of the deadline which is an absolute time. An example of these data for three tasks is shown on Figure 3.

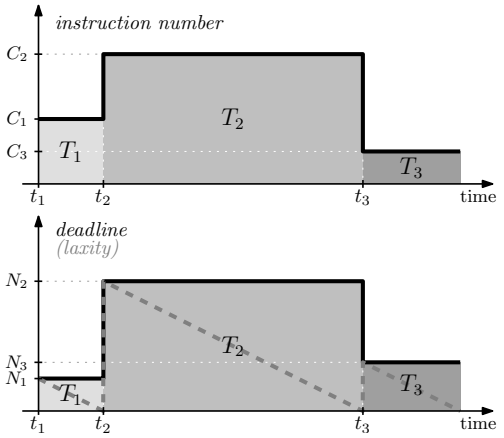


Fig. 3. Set points sent by the operating system for each task $T_i$: the instruction number $C_i$ and the deadline $N_i$ (or the laxity $L_i$)

### A. Speed Set Point Building

An intuitive speed set point is the average one: for each task $T_i$, the average reference is the ratio between the instruction number to compute and the time to do it, i.e. $C_i/N_i$. But this intuitive set point is not energy efficient. Indeed, on Figure 4 one can see the maximum computational speed when the system is running at high voltage, that is $\omega^{max} = \sigma \cdot F_{V_{high}max} \cdot V_{high}$ from (2), and respectively the maximum possible speed at low voltage, that is $\omega_{max} = \sigma \cdot F_{V_{low}max} \cdot V_{low}$. Note that we will explain in section IV how to compute $\sigma$, thanks to (5). It is easy to imagine that for all tasks with an average speed set point upper than $\omega_{max}$, the system will run at high supply voltage. Therefore it will

consume a lot because of the (quasi)-quadratic relationship between the supply voltage and the energy consumption.

A solution to avoid running the whole task at the penalizing high supply voltage as soon as the average speed set point is higher than $\omega_{max}$ is to schedule the tasks, as shown on Figure 4 (bottom plot):

- If the average speed set point of the current task $T_i$ is lower than $\omega_{max}$, the system could run at $V_{low}$ with this average speed set point, as for the tasks $T_1$ and $T_3$.
- On the other hand if the average speed set point overshoots $\omega_{max}$, as for the task $T_2$, the system has to run at $V_{high}$ to perform the task before its deadline. However, instead of executing the whole task with the high supply voltage (as done with the intuitive method), the task can be executed at $V_{high}$ during a certain time and then it would be finished at $V_{low}$. In order to minimize the high voltage running time, the system has hence to run at maximum speed when the supply voltage is $V_{high}$. Thus, after a certain high voltage and maximum speed running time, the voltage level could drop to perform the end of the task with a speed lower than $\omega_{max}$.

To sum up, an energy efficient method would perform a task by running the shortest possible time at high voltage. In other words, a task with an important computational load (such as $T_2$) will be executed at $V_{high}$ and maximal speed from its beginning ($t_2$) until a certain time ($k$). Then the task could be finished at $V_{low}$ and a speed under $\omega_{max}$, which will be enough to fit it with its deadline ($t_3$). However, the time $k$ could not be *a priori* known, therefore a predictive control law is designed to dynamically calculate it.

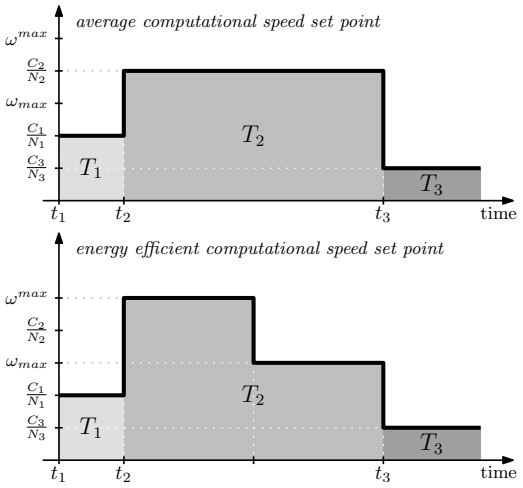


Fig. 4. Different speed set point building: an intuitive vs. an energy efficient computational speed set point behavior

The task information given by the operating system are not enough anymore and we now also need some information about system resources, such as the maximum speed for the different voltage levels ($\omega^{max}$ and $\omega_{max}$). Moreover, we need to know what it has already been done in order to predict the minimum high voltage running time, and for this reason a computational speed feedback loop is required.

### B. Fast Predictive Control

To minimize the energy consumption the system has to run the shortest possible time with the penalizing high supply voltage (and at maximum speed). Thus, the controller dynamically calculates if the system needs to run at $V_{high}$ (and at $\omega^{max}$) or if the low voltage level (and a speed lower than $\omega_{max}$) will be enough to compute the task before its deadline. This principle could be formulated as a predictive control problem: for each task $T_i$, what is the speed set point which will minimize the high voltage running time $t_{V_{high}}$ while guaranteeing that the executed instruction number is equal to the number of instructions $C_i$ to do:

$$\min\ t_{V_{high}} \quad / \quad \int_{N_i} \omega\ dt = C_i$$

Nevertheless, the speed set point can be obtained in an easier and faster way. We simply need to know *i)* what the processor has to do and *ii)* how much time is available to do it. As the speed set point is dynamically calculated, i.e. at each sampling time, the remaining time before the end of the task is necessary. This is why the laxity $L_i$ will be used instead of the deadline $N_i$.

Firstly, the predictive average speed required to perform the task exactly on its deadline, i.e. $\delta$, is calculated at each sampling time $T_s$. The value of $\delta$ can be easily described as the ratio between *what the processor has to do to compute the task* minus *what it has already done* (that is corresponding to *what it remains to do*) and *the remaining time before the end of the task*. This principle can be mathematically expressed as following (where $t_i$ is the beginning of the task $T_i$):

$$\delta(t^+) = \frac{C_i(t) - \int_{t_i}^{t-t_i} \omega(t)\ dt}{L_i(t)} \tag{3}$$

One could note that $C_i$, $N_i$ and $L_i$ are piecewise defined because they change for each task and, furthermore, the operating system could decide to modify them during the running time of a task.

The equation (3) has to be implemented and the discretization leads to (4), where $\Omega$ is the integration of the computational speed $\omega$.

$$\begin{aligned} \Omega(t_k) &= \Omega(t_{k-1}) + T_s\omega(t_k) \\ \delta(t_{k+1}) &= \frac{C_i(t_k) - \Omega(t_k)}{L_i(t_k)} \end{aligned} \tag{4}$$

A conditional instruction is added to be coherent: indeed, the computational speed $\omega$ is integrated on the running time of each task and so when a task is executed, i.e. in the last sampling time before its deadline, the variable $\Omega$ is reset. More precisely, $\Omega$ is not set to zero to prevent the case when the task is not completely executed at its deadline. For this reason we adjust $\Omega$ with the difference between what it has already been done and what it was required to do:

$$\text{if } L_i(t_k) \leq T_s, \quad \Omega(t_k) = \Omega(t_k) - C_i(t_k)$$

The set point is finally deduced from the value of $\delta$:

- If the predictive average speed is higher than the maximum speed at low voltage, i.e. $\delta > \omega_{max}$, then the system has to go to the high supply voltage and so to the maximum speed.
- On the other hand, as soon as $\delta$ becomes lower than $\omega_{max}$, the system could switch to the low voltage level because it will be able to finish the task without going back to $V_{high}$ (if the instruction number and/or the deadline do not change). Regarding the speed set point, the $\delta$ value will allow to fit the task with its deadline.

The following equation summarizes this behavior:

$$\omega_{sp}(t_k) = \begin{cases} \omega^{max} & \text{if } \delta(t_{k+1}) > \omega_{max} \\ \delta(t_{k+1}) & \text{otherwise} \end{cases}$$

Thereby, an energy efficient computational speed set point is achieved with this control law. Moreover, one can guarantee that the number of instructions to do will be done because, if the system is slower than required, $\omega_{sp}$ will be adjusted thanks to the measurement speed feedback loop.

## IV. FREQUENCY AND VOLTAGE LEVEL CONTROL

The aim of this second part of the controller is to track the speed set point $\omega_{sp}$ by adapting the frequency $f$ and the voltage level $V_{level}$.

Regarding the frequency, as $\omega_{sp}$ is already adjusted to what it has been done, the simplest set point tracking will be enough. The controller hence needs an integral at least to guarantee a null static error. We chose the discret time controller described by (5), where $K$ is a gain.

$$\begin{aligned} \varepsilon(t_k) &= \omega_{sp}(t_k) - \omega(t_k) \\ \sigma(t_k) &= \frac{\omega(t_k)}{f(t_{k-1})V_{dd}(t_k)} \\ f(t_k) &= f(t_{k-1}) + T_s\sigma(t_k)K\varepsilon(t_k) \end{aligned} \tag{5}$$

Note that the system model can be approximated as a variable gain $\sigma$, as shown by (2). We propose to use it in the frequency control law because in practice, nothing more than this gain can be measured - when measured - due to space and time dispersion of behavior. Moreover, to simplify the identification, we simply use the previous calculated frequency $f(t_{k-1})$.

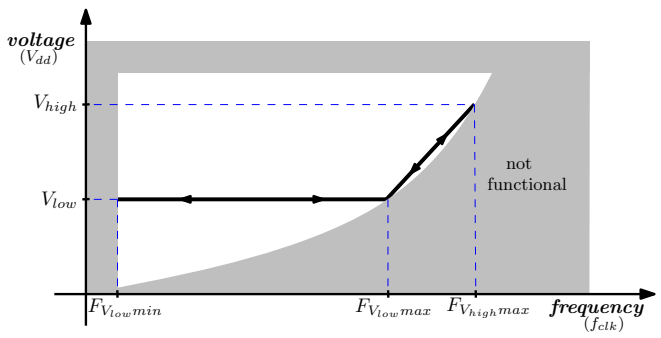


Fig. 5. Hysteresis behavior of the voltage $V_{dd}$ and the frequency $f_{clk}$

The voltage level is then deduced from the frequency according to the hysteresis behavior represented on Figure 5: $V_{level}$ can change only when the frequency is larger than the maximum frequency at low voltage level, i.e. $F_{V_{low}max}$. So if the frequency is increasing then the voltage level is set to the high level, i.e. $level_{high}$ (in order that the Vdd-hopping increases the supply voltage), and respectively $level_{low}$ if the frequency is decreasing:

$$\begin{aligned} &\Delta f = f(t_k) - f(t_{k-1}) \\ &\text{if } f(t_k) > F_{V_{low}max} \\ &\quad V_{level}(t_k) = \begin{cases} level_{high} & \text{if } \Delta f > 0 \\ level_{low} & \text{if } \Delta f < 0 \end{cases} \\ &\text{else} \\ &\quad V_{level}(t_k) = V_{level}(t_{k-1}) \\ &\text{end} \end{aligned}$$

Moreover, when the supply voltage is higher than $V_{low}$ then $f$ is forced to the maximum possible frequency (approximated by a linear function on the hysteresis):

$$\begin{aligned} &f(t_k) = a \cdot V_{dd}(t_k) + b \quad \text{if } V_{dd}(t_k) > V_{low} \\ &\text{with} \quad \left| \begin{aligned} a &= \frac{F_{V_{high}max} - F_{V_{low}max}}{V_{high} - V_{low}} \\ b &= F_{V_{high}max} - a \cdot V_{high} \end{aligned} \right. \end{aligned} \tag{6}$$

This last step is needed in order to *i)* minimize the energy consumption by reducing the high voltage running time and *ii)* guarantee the critical path by controlling the frequency during transitions to not go to the *not functional* area.

## V. PERFORMANCE EVALUATION

This section presents the simulation results of the energy controller depicted on Figure 1. Three tasks to execute are proposed for the benchmark test: the first task starts with 4 instructions to do in $0.5\mu s$, then a 65 instruction task has to be executed in $2.5\mu s$ and the last one has to compute 10 instructions in $1\mu s$. These data (usually provided by the operating system) are shown on Figure 6. Note that the laxity could be simply built as following: at the beginning of the task, the laxity is equal to the deadline and then, at each sampling interval the laxity is decreased of the value of the sampling time $T_s$ in order to be null at the end of the task.

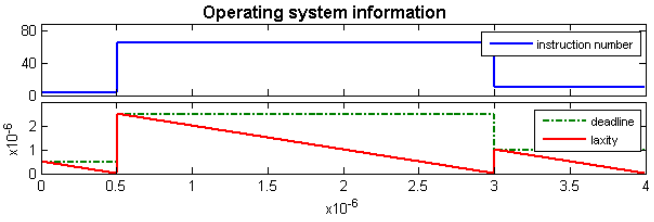


Fig. 6. References used for the simulation : number of instructions $C_i$, deadline $N_i$ and laxity $L_i$

The simulation results are shown on Figure 7. The top plot shows the average speed set point (for guideline), the speed set point $\omega_{sp}$ calculated by the computational speed controller and the measured speed $\omega$. The bottom plot shows

the supply voltage $V_{dd}$. One can verify that $\omega$ and $V_{dd}$ are proportional when the voltage is upper than $V_{low}$, due to (6).

The clock frequency $f_{clk}$, the frequency $f$ and the voltage level $V_{level}$ are not plotted because they do not provide relevant information: the frequencies are proportional to the speed and the level can be deduced from the voltage (as soon as the supply voltage increases, respectively decreases, the voltage level goes to $level_{high}$, respectively $level_{low}$).

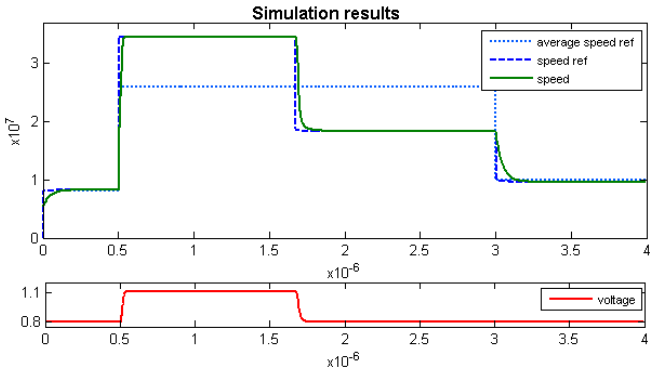


Fig. 7. Simulation results: energy consumption of $1.24 \cdot 10^{-5} J$ and computational needs of $1.43 \cdot 10^5 flops$, that is $20\%$ of energy consumption less and $10\%$ of computational needs more than a controller without DVS

The results are quantified in term of energy consumption and computational needs:

***Energy consumption of the system:*** The energy consumption is calculated in order to have an idea of the reduction achieved thanks to our proposal. Thus, the relation (1) is used and a ratio of this power consumption is added due to the Vdd-hopping principle: $20\%$ more during the voltage transition time and $3\%$ more during the steady state [10]. Finally, an integration during the whole running time gives the total energy consumption.

***Computational needs of the controller:*** The control law is compared in term of computational needs, i.e. the number of instructions required to calculated the computational speed set point $\omega_{sp}$, the frequency $f$ and the voltage level $V_{level}$. To do that, we use the Lightspeed Matlab toolbox proposed by T. Minka [11], which provides a number of flops for each instruction.

The controller is compared with a system without Dynamic Voltage Scaling (DVS): in this case the intuitive average speed set point building principle (depicted in section III) is used and the supply voltage is fixed to the penalizing high voltage, i.e. $V_{level} = level_{high}$. However, the frequency is controlled in order to guarantee that what the system has to do is really done, which implies a more complex control than ours with (5) because there is no feedback loop for the computational speed controller: a double integration is hence required to track the integral of the error.

Finally, the system runs during more than $60\%$ of the simulation time at low voltage and a reduction of the energy consumption of about $20\%$ is achieved. Moreover, the control law requires a low computational needs because the number of flops is only $10\%$ more than for a system without DVS mechanism (because of its complex frequency control law).

Note that the computational speed set point adapts itself regarding what it was really done. Indeed, one can see at time $1.8\mu s$ and $3\mu s$ on Figure 7 that the set point $\omega_{sp}$ decreases because the measured speed $\omega$ is upper than required during the voltage transitions. For this reason, the controller adapts itself to a variation of the reference, such as a modification of the deadline shown on Figures 8 and 9 or a modification of the number of instructions shown on Figure 10.

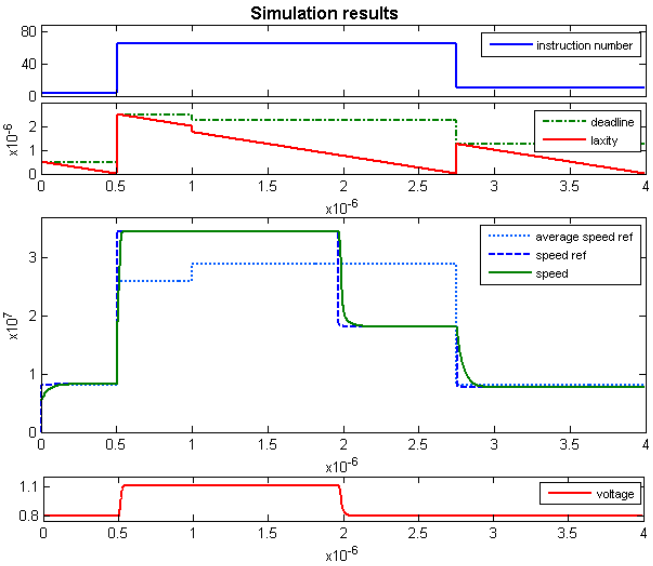


Fig. 8. Simulation results: modification of the deadline set point of the second task (before switching to low voltage): at time $1\mu s$, it finally remains $1.75\mu s$ to treat the task instead of $2\mu s$ as initially planned

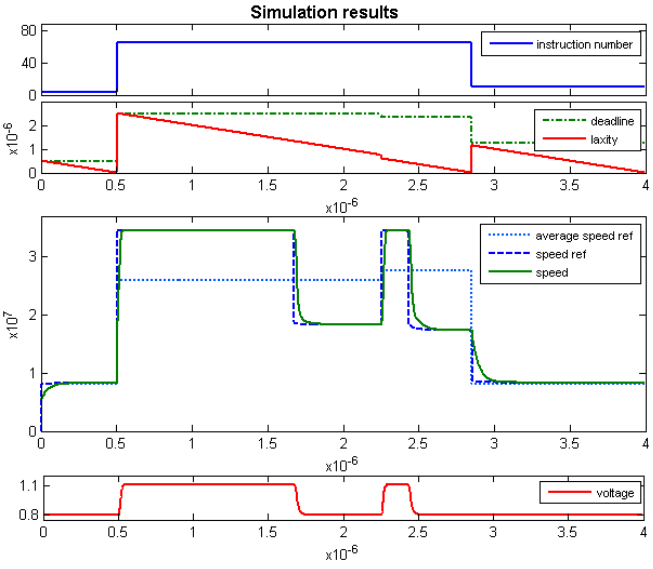


Fig. 9. Simulation results: modification of the deadline set point of the second task (after switching to low voltage): at time $2.25\mu s$, it finally remains $0.6\mu s$ to treat the task instead of $0.75\mu s$ as initially planned

Furthermore, the controller is robust to the process variability: when the device does not work well it runs more

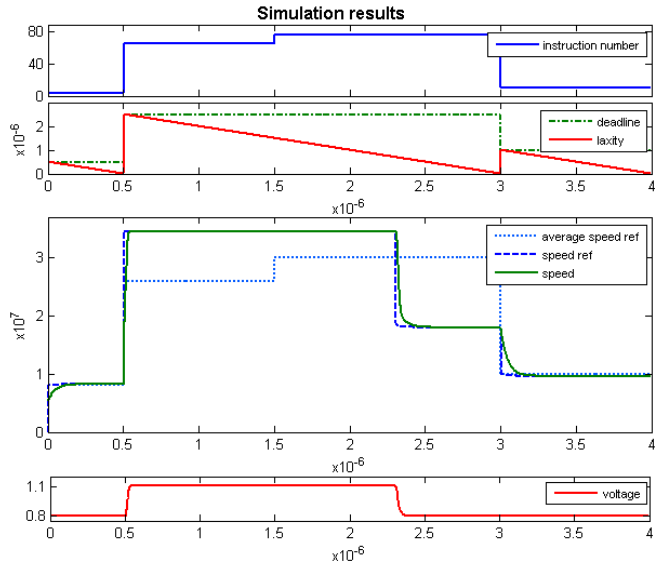


Fig. 10. Simulation results: modification of the instruction number set point of the second task during its running time: at time $1.5\mu s$, there are finally 10 instructions more than planned to treat

slowly. This leads to reduce the gains used in the equation of the device, i.e. $\omega = \alpha \cdot f_{clk} + \beta$ (see section II for further details), and the maximum speeds $\omega^{max}$ and $\omega_{max}$ are hence decreased. The simulation results on Figure 11 show how the system is still working for different ratios of $\alpha$ and $\beta$.

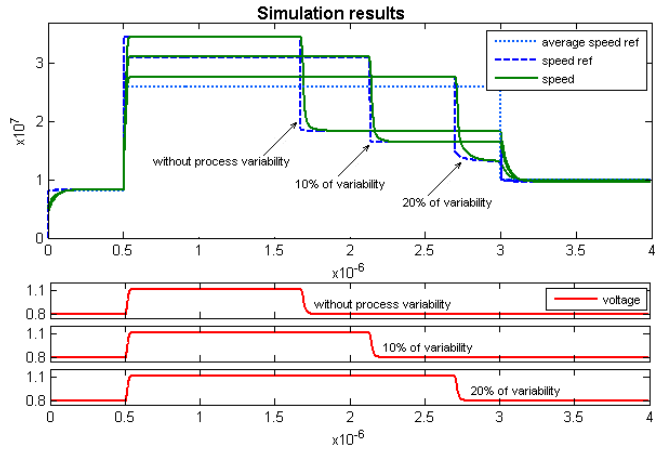


Fig. 11. Simulation results to test the robustness of the controller with 10% and 20% of process variability

## VI. CONCLUSIONS AND FUTURE WORKS

This paper proposes an architecture to control a two voltage level electronic device with a power/performance tradeoff. A fast predictive control technic gives the best computational speed set point to apply in order to minimize the penalizing high voltage running time. Then, the clock frequency and the supply voltage are controlled together in order that the measured speed tracks this set point while guaranteeing the computational performance and the critical path. The control strategy is robust and gives an important reduction of the energy consumption and a low computational needs in comparison with a system without DVS mechanism.

Next steps in this research is to test this controller in practice and develop different control strategies.

## VII. ACKNOWLEDGMENTS

This research has been supported by the NeCS Project-Team (INRIA, GIPSA-lab, CNRS) in the ARAVIS project context. ARAVIS project is a Minalogic project gathering ST Microelectonics with academic partners of different fields, namely TIMA and CEA-LETI for micro-electronics and INRIA for operating system and control. The aim of the project is to overcome the barrier of subscale technologies (45nm and smaller).

## REFERENCES

[1] C. Albea, C. Canudas de Wit, and F. Gordillo. Control and stability analysis for the vdd-hopping mechanism. In *Proceedings of the IEEE Conference on Control and Applications*, 2009.

[2] T. Burd and R. Brodersen. Processor design for portable systems. In *The Journal of VLSI Signal Processing*, volume 13, pages 203–221, 1996.

[3] T. Burd, T. Pering, A. Stratakos, and R. Brodersen. A dynamic voltage scaled microprocessor system. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, volume 35, pages 1571–1580, 2000.

[4] A. Chandrakasan and R. Brodersen. Minimizing power consumption in digital cmos circuits. In *Proceedings of the IEEE*, volume 83, pages 498–523, 1995.

[5] S. Dhar and D. Maksimovic. Switching regulator with dynamically adjustable supply voltage for low power vlsi. In *Proceedings of the 27th Conference of the IEEE Industrial Electronics Society (IECON)*, volume 3, pages 1874–1879, 2001.

[6] S. Dhar, D. Maksimović, and B. Kranzen. Closed-loop adaptive voltage scaling controller for standard-cell asics. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 103–107, 2002.

[7] S. Durand and N. Marchand. Dispositif de commande d'alimentation d'un calculateur, 2009.

[8] S. Fairbanks and S. Moore. Analog micropipeline rings for high precision timing. In *Proceeding of the International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 41–50, 2004.

[9] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Proceedings of the International Sympsonium on Low Power Electronics and Design*, pages 197–202, 1998.

[10] S. Miermont, P. Vivet, and M. Renaudin. A power supply selector for energy- and area -efficient local dynamic voltage scaling. In *PATMOS'07: 17th International Workshop on Power and Timing Modeling, Optimization and Simulation*, pages 556–565, 2007.

[11] T. Minka. The lightspeed matlab toolbox v2.2. http://research.microsoft.com/˜minka/software/lightspeed/.

[12] T. Pering, T. Burd, and R. Brodersen. Voltage scheduling in the lparm microprocessor system. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, pages 96–101, 2000.

[13] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 251–259, 2001.

[14] A. Varma, B. Ganesh, M. Sen, S. Choudhury, L. Srinivasan, and J. Bruce. A control-theoretic approach to dynamic voltage scheduling. In *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pages 255–266, 2003.

[15] H. Zakaria, L. Fesquet, S. Durand, C. Albea-Sanchez, Y. Thonnart, C. Canudas de Wit, and N. Marchand. Integrated asynchronous regulation for nanometric technologies: Application to an embedded parallel system. MINATEC CROSSROADS, 2008.